SI Appendix

The supporting information is organized as follows:

1. A detailed description of the five model systems

We used standard genetic algorithms to evolve four well-studied computational network models and a well-studied structural model of RNA. The settings of the simulations were as follows: A population of N_{pop} individuals was initialized to random binary genomes of length B bits (random nucleotide sequences of length B bases in the case of RNA). In each generation all of the individuals in the population were evaluated, and the top L circuits passed unchanged to the next generation [elite strategy (1)]. The L worst circuits were replaced by a new copy of all of the elite circuits. Pairs of circuits were recombined (using crossover probability of P_c), and then each circuit was randomly mutated (mutation probability P_m per genome). Each experiment was run for a maximal number of generations (G_{max}). A simulation could terminate in less than G_{max} generations, if the fitness threshold (*TH*) was achieved for the goal (or for all goals in case of MVG evolution).

1.1 Combinatorial logic circuits composed of NAND gates (model 1)

Genome description. A binary genome of B = 265 bits, composed of 27 genes, which coded for 26 2-input NAND gates and a single output interface (270 bits and 28 genes, in the case of 2-output circuits). Each gene on the genome was composed of two input fields encoding for the input connection. Genome and genotype-phenotype mapping are described in SI Fig. 6.

Fitness calculation. Goals were Boolean functions. Each goal was defined by a 6-input truth table (with $2^6 = 64$ entries). Each circuit in the population was evaluated on all possible input values combinations. The fitness was the relative number of correct entries in the truth table. Thus, a perfect circuit had fitness = 1. A fitness penalty of 0.05 was given for every additional gate above a predefined number of effective gates (21 gates for the simulations in the paper), where we define 'effective gates' as gates with a directed path to the output.

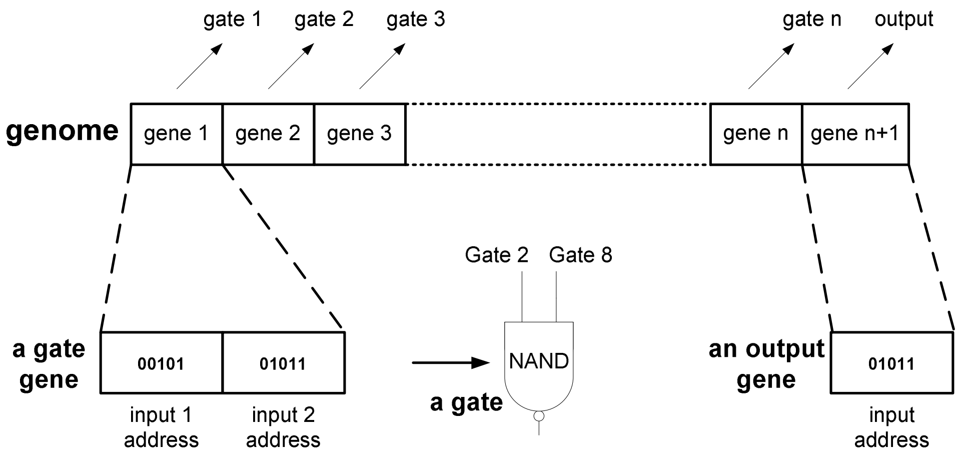


Figure 6. Model 1 genome description. The binary genome is composed of $n + 1$ genes: n genes that code for gates (here, $n = 26$) and a single gene that codes for the output. Inputs address codes were translated as follows: $0.5 \geq$ inputs (x,y,z,w,p,q), $6.31 \geq$ gates 1.26.

Varying goals scenarios. Three scenarios were simulated:

(*i*) The goal changed between eight 6-input 1-output goals G1-G8 of the form G = F(M1,M2,M3) where M1 = *x* XOR *y*, M2 = *w* XOR *z* and M3 = *p* XOR q. The goals switched in a probabilistic manner as a random walk on a graph (see SI Fig. 7). Epoch time (time between goal switches) was E = 20 generations.

G1 = (M1 AND M2) AND (M2 AND M3)

G2 = (M1 AND M2) AND (M2 OR M3)

G3 = (M1 OR M2) AND (M2 AND M3)

G4 = (M1 OR M2) AND (M2 OR M3)

G5 = (M1 AND M2) OR (M2 AND M3)

G6 = (M1 AND M2) OR (M2 OR M3)

G7 = (M1 OR M2) OR (M2 AND M3)

G8 = (M1 OR M2) OR (M2 OR M3)

Note that each switch imposed a change of a single 'module' in the goal. The function F is thus composed of 3 'modules' (combinations of AND and OR operations).

(*ii*) The goal changed between four 6-input 2-output goals G9-G12 of the form G = {F1(M1,M2,M3); F2(M1,M2,M3)}. M1, M2 and M3 defined as in (i).The goals switched in a probabilistic manner as a random walk on a graph (see SI Fig. 7). Epoch time was E = 20 generations.

G9 = {M1 AND M2; M2 AND M3}

G10 = {M1 AND M2; M2 OR M3}

G11 = {M1 OR M2; M2 AND M3}

G12 = {M1 OR M2; M2 OR M3}

Note that each switch imposed a change of a single ‘module’ in the goal.

(*iii*) Goals G13-G16 where defined similarly to (ii), but with M1 = x EQ y, M2 = w EQ z and M3 = p EQ q (where EQ is the equal logic function).

Genetic algorithm settings. $B = 265$; $N_{\mathrm{pop}} = 2000$; $L = 500$; $P_{\mathrm{c}} = 0.5$; $P_{\mathrm{m}} = 0.5$; $G_{\max} = 3 \times 10^6$; $TH = 1$.

Evolution time estimation. T_{FG} estimation was based on 30 experiments for each of the 16 goals. T_{MVG} estimation was based on 30 experiments for each of the three MVG scenarios (i-iii). T_{RVG} estimation was based on 30 experiments for each one of the randomly varying goals scenario.

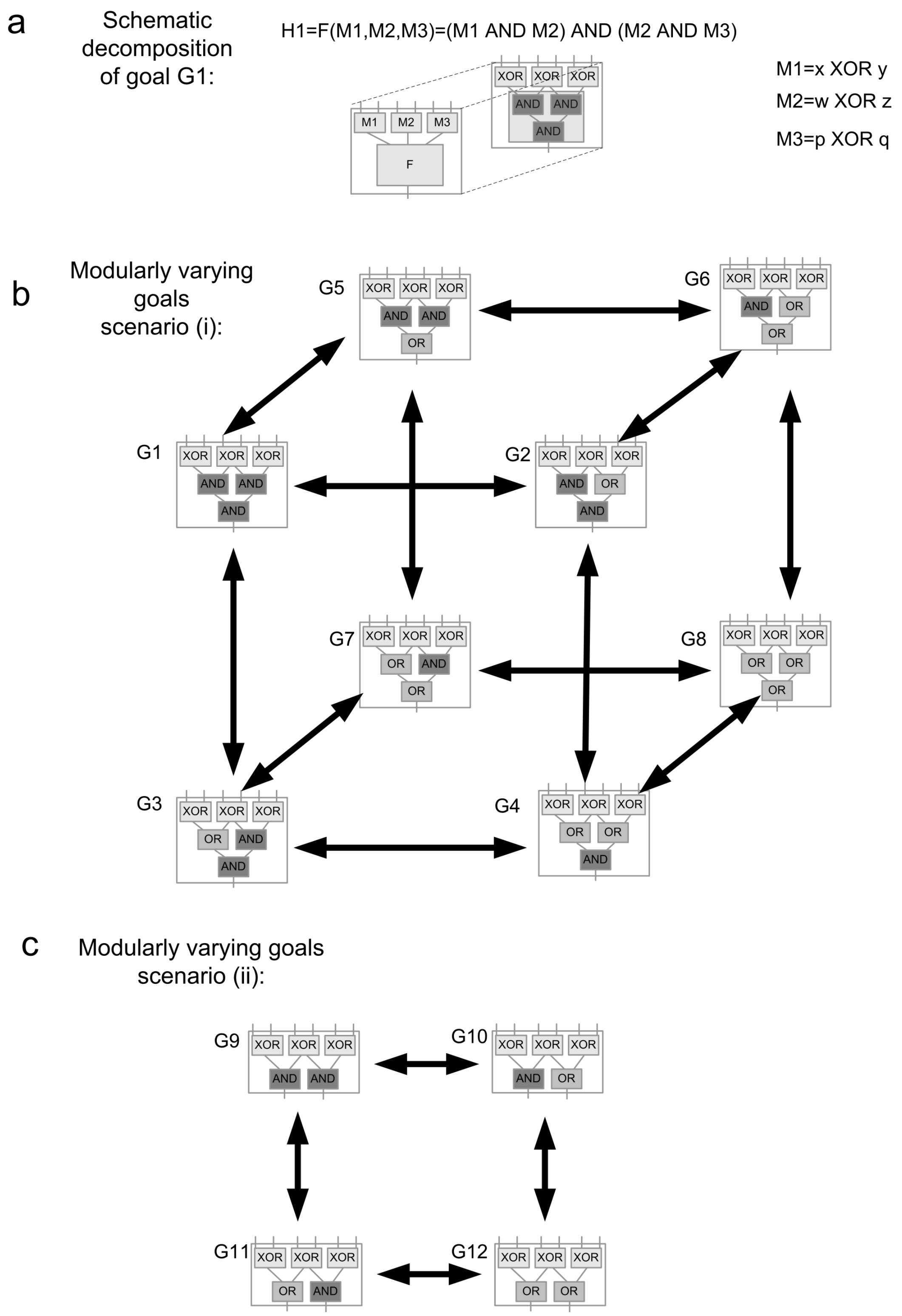

a
Schematic decomposition of goal G1:
H1=F(M1,M2,M3)=(M1 AND M2) AND (M2 AND M3)
M1=x XOR y
M2=w XOR z
M3=p XOR q
M1 M2 M3
F
XOR XOR XOR
AND AND
AND
b
Modularly varying goals scenario (i):
G1
G2
G3
G4
G5
G6
G7
G8
c
Modularly varying goals scenario (ii):
G9
G10
G11
G12

Fig. 7. Modularly varying goals in model 1 (Logic circuits). (*a*) A schematic decomposition of the goal G1 (example of evolved circuits is shown in Fig. S3). (*b*) Scenario (*i*): 6-input 1-output goals. The goal switched during evolution in a

probabilistic manner as a random walk on the 8-node graph. For example if the previous goal was G1, the next goal is randomly chosen to be one of its three neighbors: G2, G3, or G5. Note that every switch a single AND module is changed to OR or vice versa. (*c*) Scenario (*ii*): 6-input 2-output goals. The goal switched during evolution in a probabilistic manner as a random walk on the 4-node graph. Scenario (*iii*) is similar to (*ii*) but with EQs instead of XORs.

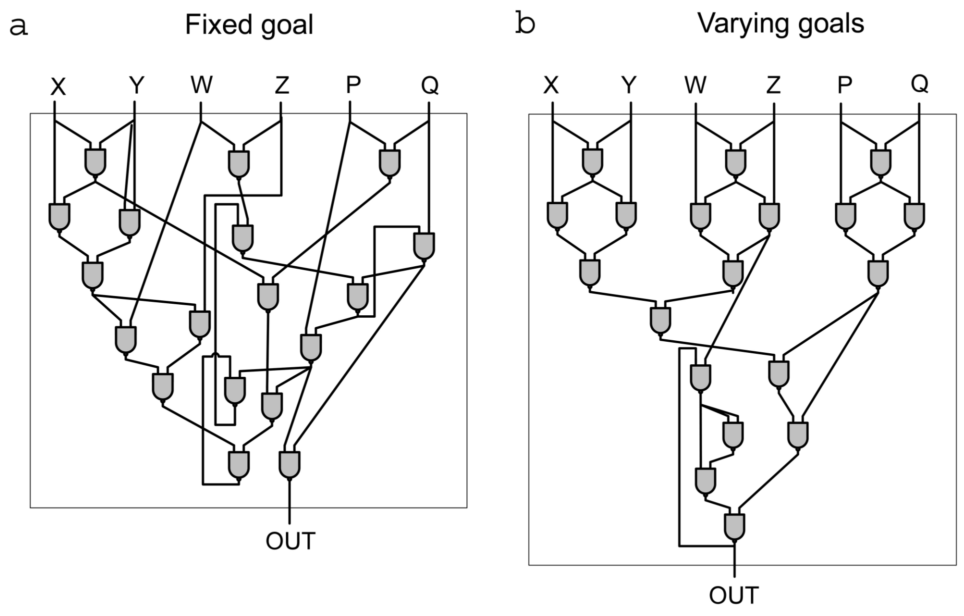


Fig. 8. Examples of circuits evolved under fixed and modularly varying goals. (*a*) A circuit composed of NAND gates evolved under fixed goal evolution. The 6-input goal was G1 = F(M1,M2,M3) = (M1 AND M2) AND (M2 AND M3) where M1 = *x* XOR *y*, M2 = *w* XOR *z* and M3 = *p* XOR *q*. (*c*) A circuit evolved under MVG that solves goal G1 [using scenario (*i*)]. Note that the circuits evolved under MVG have modular structure, with one module corresponding to each of the subproblems shared by the varying goals (XOR in this case) (2).

1.2 Feed-forward combinatorial logic circuits composed of several gate types (model 2)

Genome description. A binary genome of B = 112 bits composed of 15 genes (108 bits and 14 genes for the 2-ouput version), which coded for fifteen 2-inputs logic gates arranged in 4 layers with 8,4,2,1 gates (3 layers with 8,4,2 gates for the 2-output version). Connections were from each layer to the next layer only. The output was defined as the output of the gate(s) in the last layer . Each gene was composed of three fields: one encoded for the gate type (AND, OR or NAND) and two fields encoded for the input connections. Genome and genotype-phenotype mapping are described in SI Fig. 9.

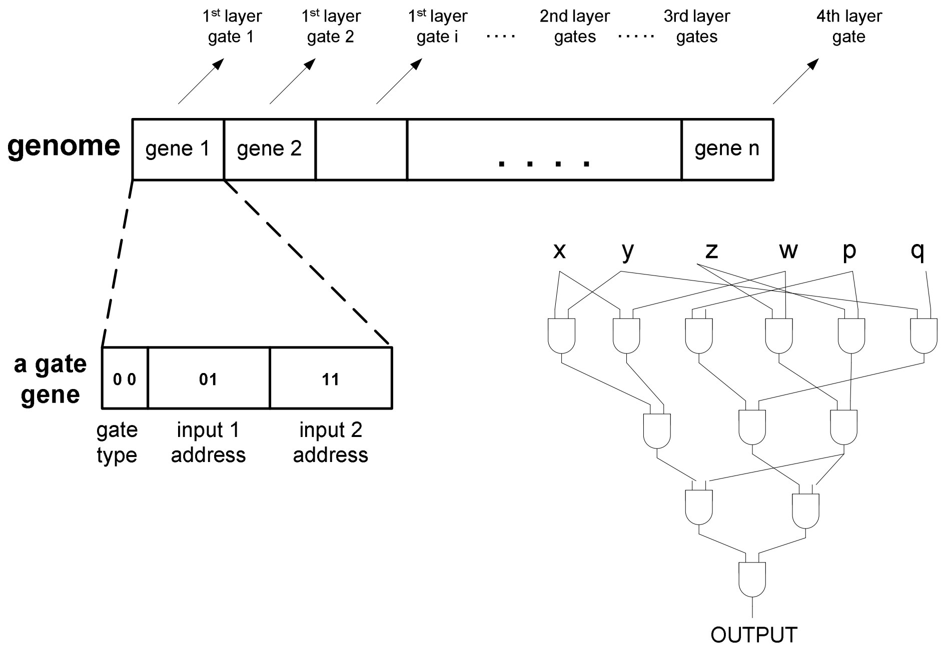


Fig. 9. Genome description (model 2 and model 4). The binary genome is composed of 15 genes. Each gene codes for a gate. The gates were arranged in a feed-forward manner in four layers. Each gene had three fields: a field that encoded the gate type and two input fields encoding for the inputs using the index of the gate in the previous layer. The gate type in model 2 was of 2 bits with the following mapping: 00: AND; 01: OR; 10: NAND; 11: AND. The gate type in model 4 was of 2 bits in the following mapping: 00 :xy; 01: $x + y - xy$; 10: $1 - xy$; 11: xy. Output was defined as the 4th layer single gate output.

Fitness calculation. Goals were Boolean functions. Each goal was defined by a 6-input truth table (with $2^6 = 64$ entries). Each circuit in the population was evaluated on all possible input values. The fitness was the relative number of correct entries in the truth table. A perfect circuit thus had fitness = 1. A fitness penalty of 0.05 was given for each additional gate above 12 gates (for the 1-output goals) or 11 gates (for the 2-outputs goals).

Genetic algorithm settings. $B = 112$; $N_{\text{pop}} = 700$; $L = 200$; $P_c = 0.5$; $P_m = 0.7$; $G_{\max} = 10^6$; $TH = 1$.

Evolution time estimation. T_{FG} estimation was based on 30 experiments for each of the different 12 goals. T_{MVG} estimation was based on 50 experiments for each of the scenarios. T_{RVG} was based on 30 experiments for each of the goals under each scenario.

Feed-forward combinatorial logic circuits composed of several gate types (model 2 - small version)

This model was studied mainly because its genome is small enough to allow a complete mapping of the fitness landscape, as described in the main text.

Genome description. A binary genome of 38 bits, composed of 7 genes, which coded for seven 2-inputs logic gates arranged in three layers with 4,2,1 gates. Connections were from each layer to the next layer only. For the results described in the paper the input connections of the last gate were fixed (connected to the outputs of the two gates in the previous layer).

Fitness calculation. Goals were Boolean functions. Each goal was defined by a 4-input truth table (with $2^4 = 16$ entries).

Genetic algorithm settings. $B = 38$; $N_{\text{pop}} = 100$; $L = 30$; $P_c = 0$; $P_m = 0.7$; $G_{\max} = 10^7$ and $TH = 1$.

1.3 Integrate-and-fire neural network model (model 3)

Genome description. A binary genome of 68 bits, composed of 7 genes, coded for 7 neurons arranged in 3 layers with 4,2, and 1 neurons in a feed-forward manner. Each neuron had two inputs from the previous layer. The output of the network was the output of the single neuron in the last layer. Connections were weighted with weights in the range -2 to 2. Each neuron summed its weighted inputs and, if the sum exceeded a threshold, the neuron fired (outputted a value of one). This is equivalent to a step-like transition function. Genome and genotype-phenotype mapping are described in SI Fig. 10.

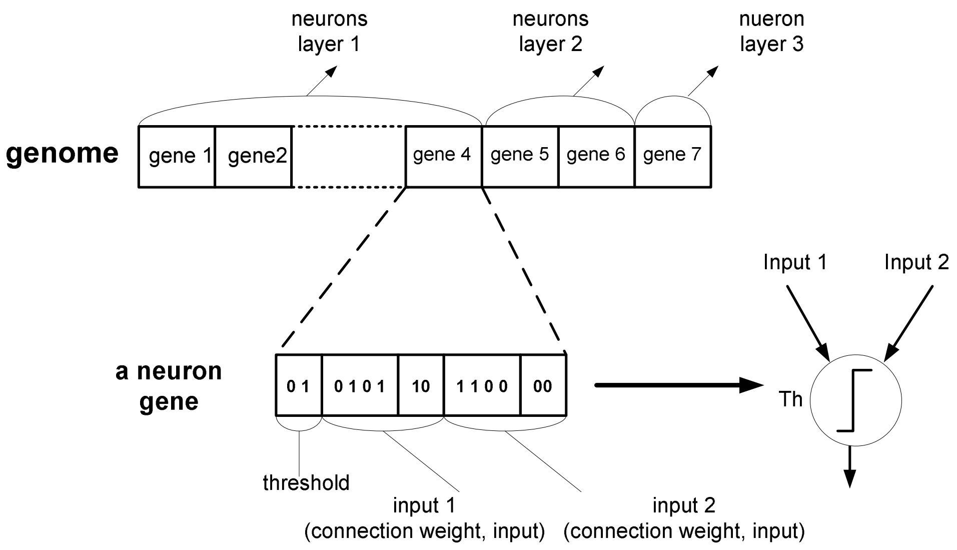


Fig. 10. Model 3 genome description. The binary genome was composed of seven genes. Each gene encoded for a neuron. The neurons were arranged in a feed-forward manner in three layers. Each gene had five fields: a field that encoded the threshold of the transition function; two fields encoded for the inputs using the index of the gate in the previous layer; two fields coded for the connection weights. The threshold field was of 2 bits in the following mapping: 00: $P_{th} = -2$; 01: $P_{th} = -1$; 10: $P_{th} = 1$; 11: $P_{th} = 0$. Each weight field was of 3 bits in the following mapping: 000: W = -2; 001: W = -1.5; 010: W = -0.5; 011: W = -1; 100: W = 2; 101: W = 1.5; 110: W = 0.5; 111: W = 1. Note that this mapping represents a smooth mapping (gray code). Output was defined to be the 3rd layer neuron output.

Goal description. We used a simple biological model for the problem of detecting combinations of morphogens levels by a cell. Neurons schematically represent signaling proteins in the cell. The proteins were arranged in 3 layers representing different cellular localizations: 4 membrane proteins that act as receptors (layer 1), 2 cytoplasmic proteins (layer 2) and a single nuclear protein - a transcription factor (layer 3). Each protein had two binding sites where proteins of previous layer were free to bind. The output of the network was determined by the activity of the transcription factor. This simple model was suggested to represent the essentials of simple signal transduction networks in proteins (3,7).

Fitness calculation. Each of the four network inputs (morphogens) could be present at three levels: low, medium and high. The goal was to decide if a specific combination of morphogen levels is satisfied (SI Fig. 11). The fitness was defined as the relative number of correct decisions over all possible combinations of morphogen levels. A 'perfect' network had fitness = 1.

Varying goals scenarios. The goals were chosen such that they could be decomposed into a combination of two subgoals S1 and S2. MVG was applied by switching between R = AND and R' = OR combination of 8 different 2-morphogen subgoals (see SI Fig. 11). The goal was switched every E = 20 generations.

Genetic algorithm settings. $B = 68$; $N_{\text{pop}} = 1000$; $L = 300$; $P\text{c} = 0.5$; $P\text{m} = 0.9$; $G_{\text{max}} = 10^7$; $TH = 0.98$.

Evolution time estimation. T_{FG} estimation was based on 30 experiments for each of the 16 different goals. T_{MVG} estimation was based on 40 experiments for each of the 8 scenarios. T_{RVG} was based on 30 experiments for each of the goals under each scenario.

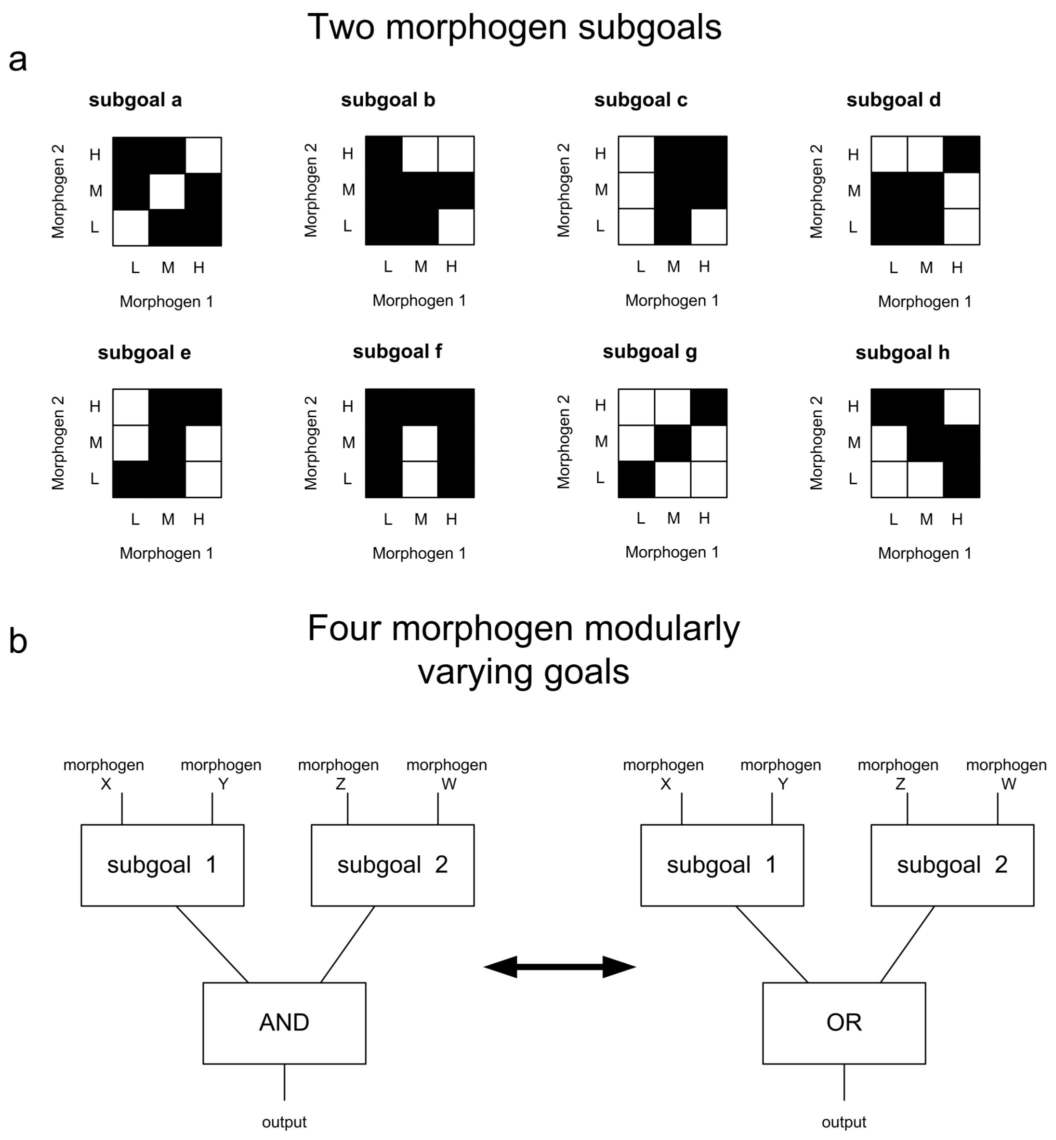


Fig. 11. Varying goals for neural network model (model 3). (*a*) Each subgoal had 2 inputs (two morphogens). Morphogen concentrations could be of three levels: Low, Medium, and High represented by L, M, and H respectively. Each subgoal (S1, S2) identifies a unique combination of the two morphogens levels; (white = 0,black = 1). (*b*) The goal was to decide whether a combination of two subgoals was identified. We used different combinations of the two 8-subgoals described in *a*, where we switched between R = AND and R’ = OR combination of subgoals every E = 20 generations.

1.4 Continuous function circuits (model 4):

Genome description. A binary genome of $B = 112$ bits composed of 15 genes (108 bits and 14 genes for the 2-ouput version), which coded for fifteen 2-inputs continuous function gates arranged in 4 layers with 8,4,2,1 gates (3 layers with 8,4,2 gates for the 2 output version). Connections were from each layer to the next layer only. The output was defined as the output of the gate(s) in the last layer. Each gene was composed of three fields: one field for the function type (of 3 possible polynomial functions: xy, $1 - xy$, $x + y – xy$, that represent continuous versions of AND, NAND and OR gates respectively) and two fields for the inputs. Genome and genotype-phenotype mapping are described in SI Fig. 9.

Fitness calculation. The goals were defined as multivariate polynomials of 6 variables x, y, w, z, p, q. For fitness evaluation each input was sampled uniformly from the interval [0,1]. For the results presented in the paper we used 4 samples per input. The fitness was defined as –one minus the mean relative distance of the outputs and the goal values (an approximation of the integral of the difference between the network computation and the goal function on the interval [0,1] of all inputs). A perfect circuit had fitness = 1. However in the current settings perfect circuits often do not exist, therefore we chose a threshold $TH = 0.98$.

Varying goals scenarios. Three scenarios were simulated:

(i) The goal changed between eight 6-input 1-output goals G1-G8 of the form G = U(T1,T2,T3). The varying goal switched in a probabilistic manner as a random walk on the graph (in a similar manner as described in SI Fig. 7). Epoch time was E = 20 generations.

G1 = (T1 · T2) · (T2 · T3)

G2 = (T1 · T2) · [T2 + T3 - T2 · T3]

G3 = (T1 + T2 - T1 · T2) · (T2 · T3)

G4 = (T1 + T2 - T1 · T2) · (T2 + T3 - T2 · T3)

G5 = T1 · T2 + T2 · T3 - [(T1 · T2) · (T2 · T3)]

G6 = T1 · T2 + (T2 + T3 - T2 · T3) - [(T1 · T2) · (T2 + T3 - T2 · T3)]

G7 = (T1 + T2 - T1 · T2) + (T2 · T3) – [(T1 + T2 - T1 · T2) · (T2 · T3)]

G8 = (T1 + T2 - T1 · T2) + (T2 + T3 -T2 · T3) – [(T1 + T2 - T1 · T2) · (T2 + T3 -T2 · T3)]

where T1 = $x + y - 2xy$, T2 = $w + z$ - 2wz and T3 = $p + q - 2pq$. Note these polynomial functions represent continuous versions of XOR functions.

Note that each switch imposed a change of a single 'module' in the goal.

(ii) The goal changed in a probabilistic manner between four 6-input 2-output goals G9-G12 of the form G = {U1(T1,T2,T3); U2(T1,T2,T3)}. The goals switched in a probabilistic manner as a random walk on the graph (similar to SI Fig. 7). Epoch time was E = 20 generations.

G9 = {T1 · T2; T2 · T3}

G10 = {T1 · T2; T2 + T3 - T2 · T3}

G11 = {T1 + T2 - T1 · T2; T2 · T3}

G12 = {T1 + T2 - T1 · T2; T2 + T3 - T2 · T3}

where T1 = $x + y - 2xy$, T2 = $w + z$ - 2wz and T3 = $p + q - 2pq$.

Note that each switch imposed a change of a single 'module' in the goal.

iii) Goals J13-J16 where defined similarly to (ii), but with T1 = $1 + 2xy - x - y$, T2 = $1 + 2wz - w - z$ and T3 = $1 + 2pq - p - q$. Note these polynomial functions represent continuous versions of EQ functions.

Genetic algorithm settings. $B = 112$; $N_{\mathrm{pop}} = 700$; $L = 200$; $P\mathrm{c} = 0.5$; $P\mathrm{m} = 0.7$; $G_{\max} = 10^5$; $TH = 0.98$.

Evolution time estimation. T_{FG} estimation was based on 30 experiments for each of the different goals. T_{MVG} estimation was based on 30 experiments for each of the scenarios. T_{RVG} estimation was based on 30 experiments for each goal under each RVG scenario.

1.5 RNA secondary structure (model 5):

Genome description. A fixed length genome of B nucleotides. Each position could be occupied by one of the four bases A, U, G, or C. We used well established tools for secondary structure prediction (4) available as part of the Vienna software package at www.tbi.univie.ac.at/RNA/.

Fitness calculation. The goal is a defined RNA secondary structure. The fitness represents the similarity of the current structure to the goal structure. We considered the minimal free energy (MFE) structure as the predicted structure of the molecule. We used a 'tree edit' distance measure as defined in the Vienna RNA package to evaluate the difference between the structure of the RNA molecule and the goal structure. The fitness is defined as $F = 1 - d/B$, where d is the distance and B is the sequence length. A fitness $F = 1$ reflects 100% similarity in structure.

Varying goals scenarios. We present here results of two MVG simulations each one with 4 goals.

(*i*) The main goal G1 was a natural tRNA secondary structure (the phenylalanine tRNA of *S. cerevisiae*, SI Fig. 12*a*). If one considers a hairpin as a structural module (4), the goal G1 includes three such modules. Modularly varying variants of G1 were obtained by changing each of the three hairpins to an open loops (SI Fig. 13). Epoch time was E = 20 generations. Note that each switch imposed a change of a single 'module' in the goal.

(*ii*) As (*i*) but the main goal was a synthetic structure of length 61 nucleotides (SI Fig. 12*b*) composed of three different hairpin-like substructures.

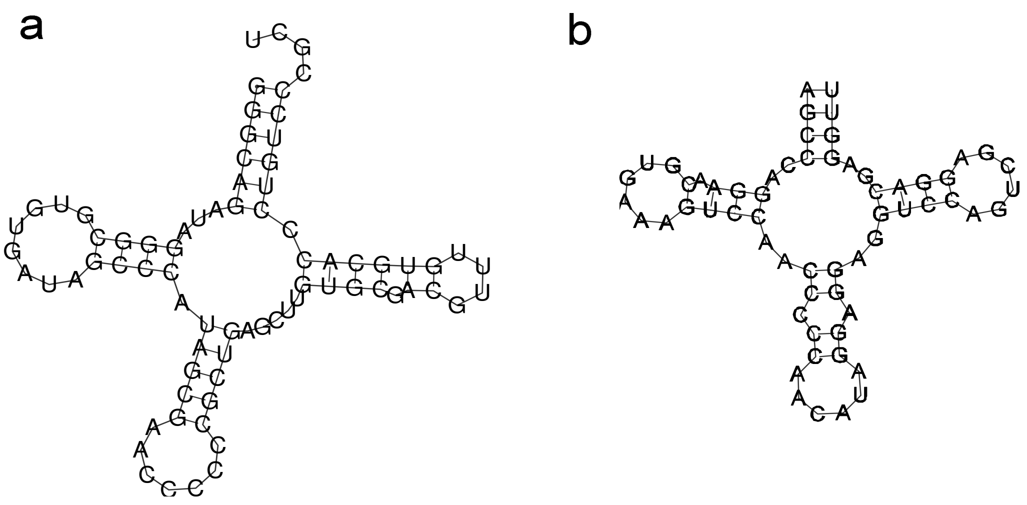


Fig. 12. Evolutionary goals defined as Secondary structure of RNA. (*a*) A phenylalanine tRNA in *S. cerevisiae*. (*b*) A synthetic structure of three different hairpin-like substructures.

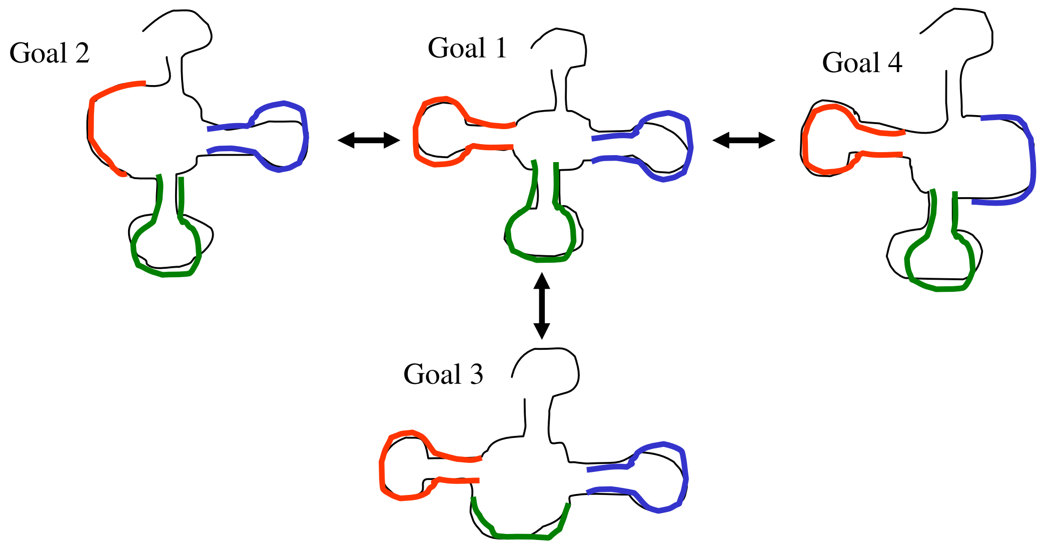


Fig. 13. Modularly varying goals in the RNA model, a schematic view. The goal switched during evolution in a probabilistic manner as a random walk on the 4-node graph. Goal 1 (G1) represents a natural secondary structure of tRNA, Goals G2,G3 and G4 are modular variants of G1.

Genetic algorithm settings. B = 76 nucleotides for scenario (i); B = 61 nucleotides for scenario (ii); N_{pop} = 500; L = 150; Pc = 0; Pm = 0.7; $G_{\max}$ = 2x10^5; TH = 1. (In model 5 we used independent mutation probabilities per locus and not per genome, thus mutation rate per locus was Pm/B).

Evolution time estimation. T_{FG} estimation was based on 40 experiments for each of the different goals. T_{MVG} estimation was based on 40 experiments for each of the scenarios. T_{RVG} estimation was based on 20 experiments for each goal under each RVG scenario.

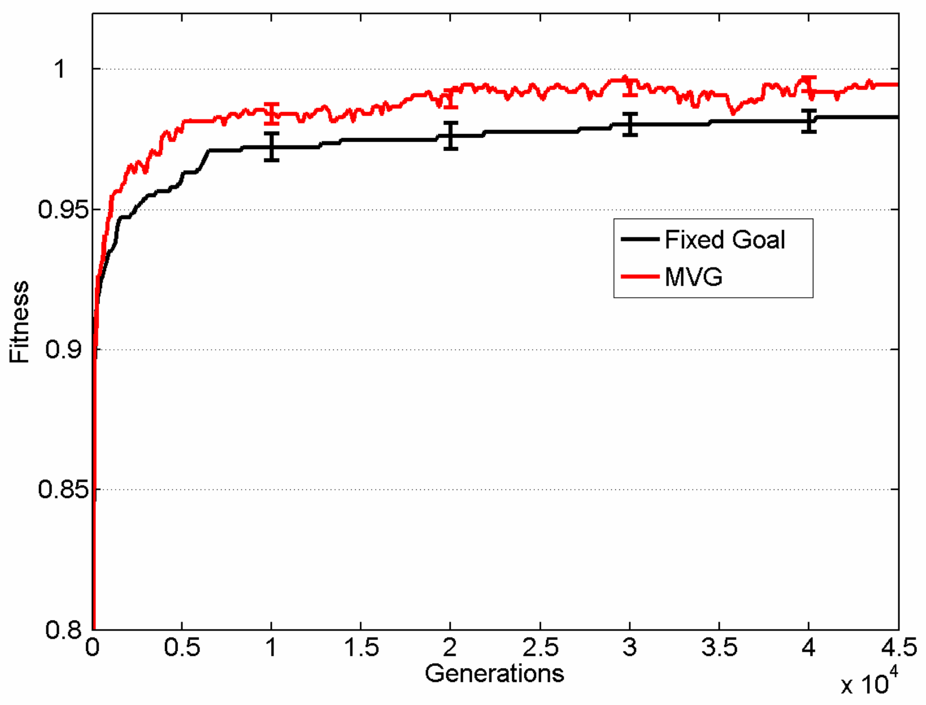


Fig. 14. Fitness through evolution in the RNA model. Maximal fitness in the population (Mean± SE) vs. generations in evolution under MVG (red) and Fixed Goal (black). The presented fitness is of goal G1 scenario i (natural tRNA). In the MVG,

the fitness presented is for epochs when the goal was the wild-type tRNA structure (G1). Fitness data are for 40 simulations in each scenario.

2. Definition of evolution time (T_{FG} and T_{MVG})

We define evolution time (*T*) as the median number of generations that is required to achieve the goal (to reach a fitness higher than a predefined threshold *TH*, where in most cases studied here *TH* = 1). To estimate *T* under fixed goal (FG) we ran the same experiment many times and calculated the median of the *T* distribution. Thus, *T* is the required number of generations such that the fitness threshold is reached with probability 0.5. If an experiment ended before the fitness threshold was achieved, *T* was considered as G_{max}. In evolution under modularly varying goals (MVG), we computed *T* for each of the different varying goals. Note that in MVG T is the total number of generations passed from the beginning of the simulation, and includes also epochs for the other goals (we do not subtract these). In RVG we referred only to the main goal, rather than the random goals. To estimate *T* and its error bars we used a bootstrap method to evaluate the median and its confidence interval (with alpha = 0.32 two sided). For the bootstrap we used 999 samples. SI Fig. 15 shows a typical behavior of fitness through evolution under fixed goal and MVG scenarios.

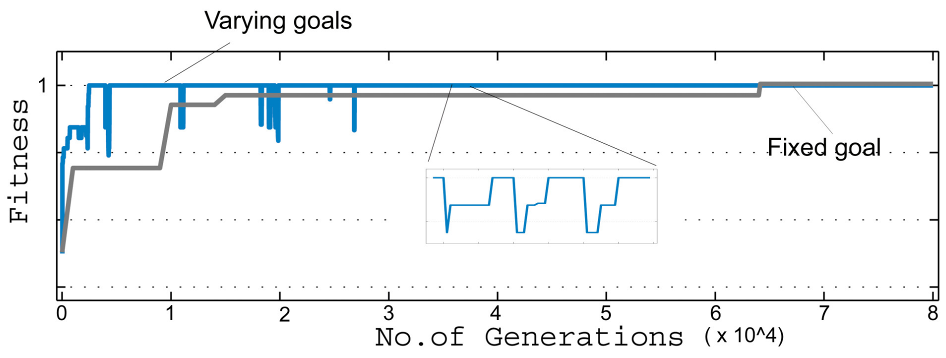


Fig. 15. Fitness under fixed and modularly varying goals. Maximal fitness in a population of networks under fixed goal (gray) and modularly varying goal evolution (blue). Evolution time is defined as the number of generations to reach fitness = 1 (or larger than a defined threshold). The goal switched every E = 20 generations. (*Inset*) Zoom into fitness around several goal switching events.

3. Speedup comparison on all scenarios of varying goals (MVG, RVG_V, RVG_C, VG_0).

SI Fig. 16 summarizes the results for speedup under all varying goals scenarios.

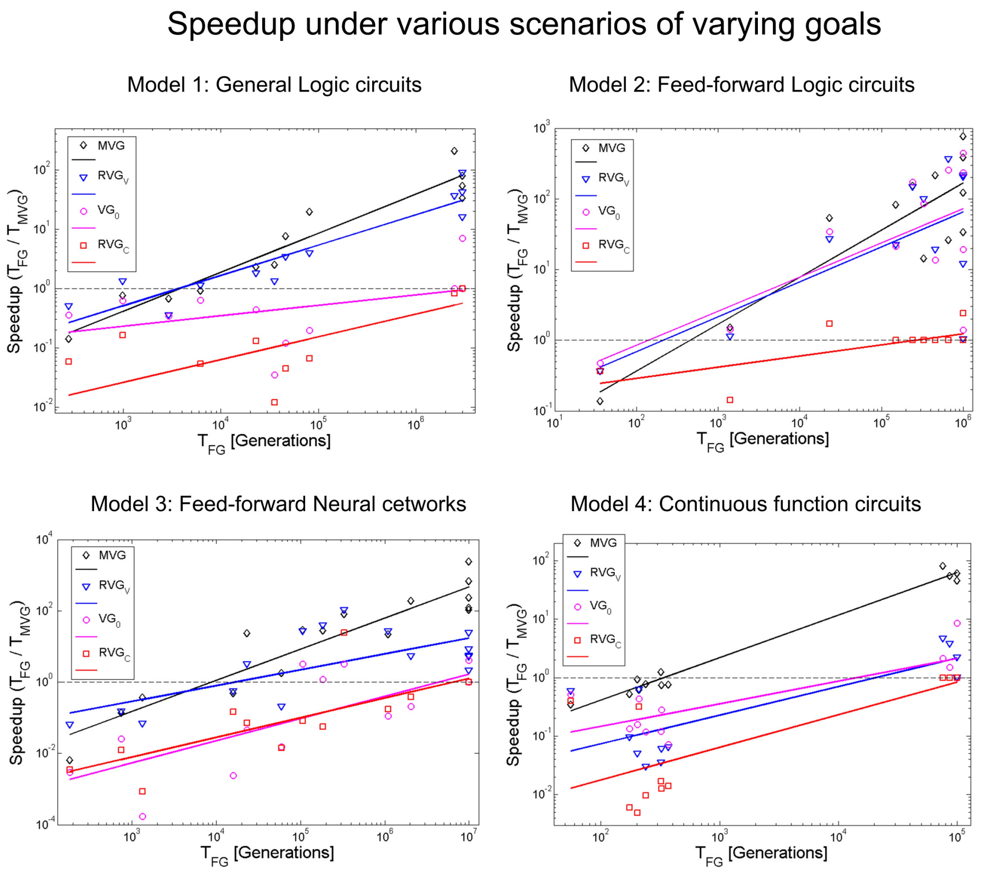


Fig. 16. Evolution speedup under varying goals. Evolution speedup under several varying goals scenarios. Points in different colors and shapes represent speedup under four different scenarios: MVG, RVG_V, VG_0, and RVG_C. Regression lines were computed using simple regression. In all scenarios the varying goal changed every 20 generations.

4. Different measures of T_{FG} and T_{MVG} and their impact on speedup measures

We found a qualitatively similar speedup using the median, mean and geometric mean as measures for evolution time. The scaling exponent for the speedup factor was also qualitatively the same (see SI Fig. 17).

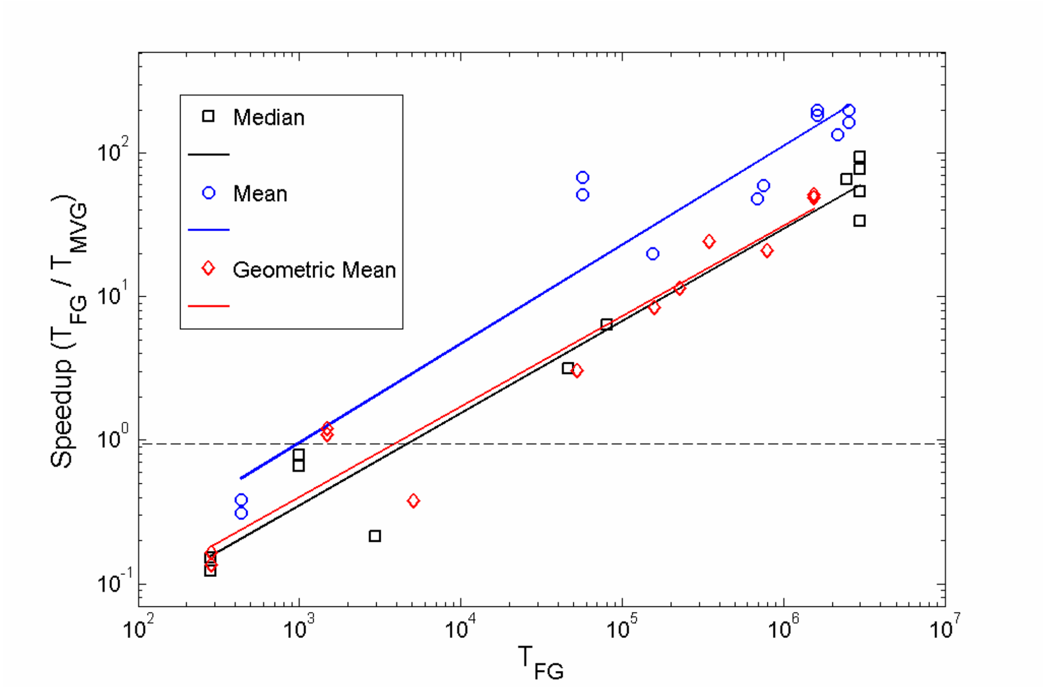


Fig. 17. Evolution Speedup with different measures for evolution time. Results are shown for model 1. Speedup scales as a power law with qualitatively similar exponents $\alpha = 0.7$ under all of the three measures.

5. Performance comparison with multiobjective optimization scenarios

We compared the speed of evolution in MVG to four different multiobjective scenarios. In the multiobjective scenarios we used the different modularly varying goals variants as the objectives, presented simultaneously (rather than varying over time as in MVG).

a) Multioutput weighted multiobjective optimization. In this scenario the network had multiple outputs, each output corresponds to a goal in the MVG scenario. The selection fitness was defined as $F = \sum_i w_i f_i$ where f_i is the fitness for output i (objective i) and w_i is its weight contribution in the aggregate fitness. We used equal weights in the simulations (for example if the number of goals was 2, $w_1 = 0.5$ and $w_2 = 0.5$).

b) Multioutput pareto multiobjective optimization. The network had multiple outputs, each output correspond to a goal in the MVG scenario (as in a.). We used pareto-based fitness assignment as proposed by Goldberg (5). In short, this method assigns equal probability of reproduction to all nondominated individuals in the population. The method works as follows: assign rank 1 to the nondominated individuals, then iteratively remove these nondominated individuals from contention and repeat assigning increasing ranks for each layer of remaining nondominated individuals. The selection fitness of each individual was defined as $1 - r / L$ where r is the rank and L is the highest rank at the current generation.

c) Single-output weighted multiobjective optimization. As in a. but the network had a single output.

d) Single-output pareto multiobjective optimization. As in b. but the network had a single output.

We compared the progress of fitness for a given goal under fixed goals (FG), MVG and all four multiobjective scenarios. For a proper comparison we compared the fitness of a specific goal under FG with its equivalent in the multiobjective scenarios in the following manner: For the multioutput cases we considered the fitness of the relevant output only. For the single output scenarios, we considered the highest fitness of the relevant goal in the population at each generation (see SI Fig. 18). By this one avoids a bias for solutions in the pareto front that have higher preference to other objectives.

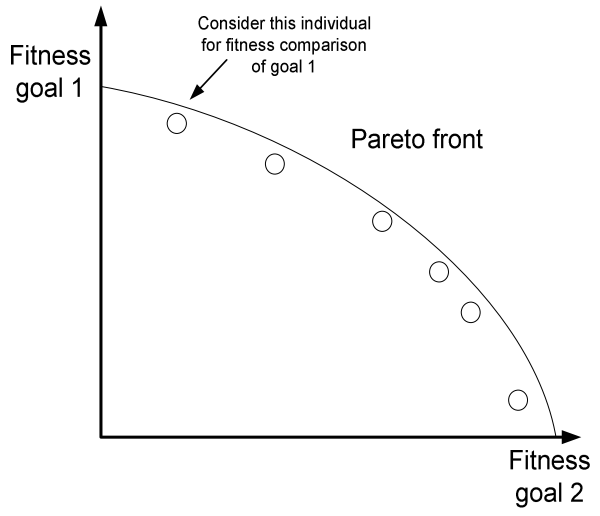


Fig. 18. Fitness extraction for a proper comparison under pareto optimization.

Genetic algorithm settings. 4-inputs 1-output version of model 1. $B = 265$; $N_{\mathrm{pop}} = 2000$; $L = 500$; $P_{\mathrm{c}} = 0.5$; $P_{\mathrm{m}} = 0.5$; $G_{\max} = 10^6$; $TH = 1$. The genome represents a pool of up to 26 2-input NAND gates. A fitness penalty of 0.1 was given for circuits with more than 11 gates (in the single output scenarios) and 14 gates (in the multioutput scenario).

Results. SI Table 2 shows results for a 4-input version of network model 1. We compared the speed of evolution for the goal G1 = (x XOR y) AND (w XOR z). The MVG scenario was composed of two goals, G1 and G2 = (x XOR y) OR (w XOR z), where the goal switched every 20 generations.

We find that MVG showed speedup, whereas all four multiobjective scenarios showed virtually no speedup. Scenario (a) was the only scenario which had $T < 10^6$ generations.

Note that the fitness of the single-output weighted multiobjective scenario saturates at fitness significantly lower than one. The two pareto scenarios seemed to plateau at fitness 0.75 for G1. The reason for this is that there exist many nondominated trivial solutions that reach fitness 0.75 for one of the goals (e.g., a network that outputs zero for all input values has fitness 0.75 for G1). Qualitatively similar results are obtained also for other models. A graph of fitness vs. generations for FG, MVG and all multiobjective scenarios is shown in SI Fig. 19 (Fig. 3 in the main text shows FG, MVG and scenario a).

Table 2. Evolution time and speedup comparison under MVG and 4 different multiobjective scenarios

Scenario		Median time (T)	Mean time (<T>)	Speedup (S)
Fixed Goal		$1.5 \times 10^4 \pm 1 \times 10^4$	$3 \times 10^5 \pm 3 \times 10^4$	-
MVG		$8 \times 10^3 \pm 3 \times 10^3$	$5 \times 10^4 \pm 1 \times 10^4$	2.2 ± 0.7
Multi Output	Weighted Multi-Objective	$1\times 10^5 \pm 4 \times 10^4$	$3 \times 10^5 \pm 6 \times 10^4$	<1
	Pareto Multi-Objective	$>10^6$	$>10^6$	<1
Single Output	Weighted Multi-Objective	$>10^6$	$>10^6$	<1
	Pareto Multi-Objective	$>10^6$	$>10^6$	<1

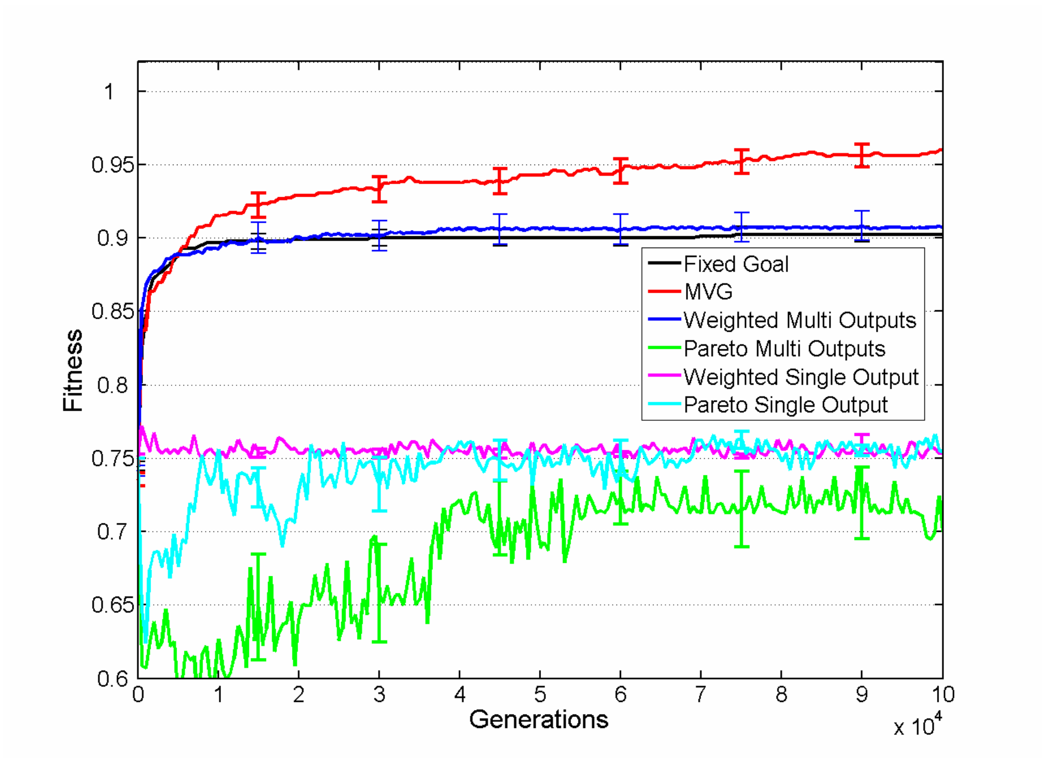
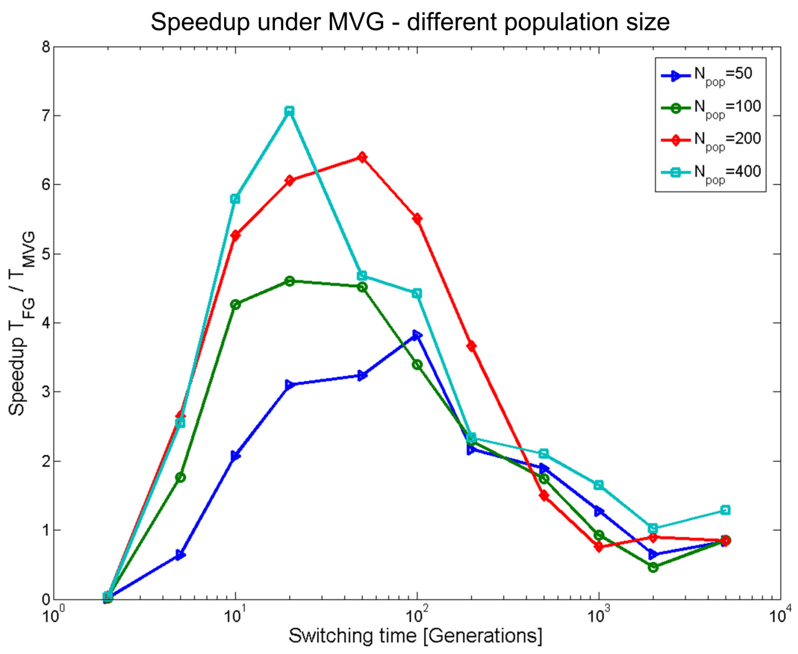


Fig. 19. Fitness as a function of time in MVG, fixed goal and all four multiobjective scenarios. Maximal fitness in the population (mean ± SE) as a function of generations for a 4-input version of model 1 for the goal G1 = (*x* XOR *y*) AND (*w* XOR *z*). For the MVG and multiobjective cases, the second goal was G2 = (*x* XOR *y*) OR (*w* XOR *z*). For MVG, data are for epochs where the goal was G1. For the multiobjective scenarios fitness is for G1 as described above. Data are averaged for 20 simulations in each case.

6. Effects of simulation parameters on the speedup

We evaluated the speedup in evolution under a wide range of simulation parameters. SI Fig. 20 shows the speedup under a range of rates of switching between goals (see also Fig. 2 in the main text), and with a different selection strategy [pure fitness-based selection as opposed to elite strategy (10)] and with a different mutation policy (mutation probability per locus and not per genome). The speedup is smaller than the results presented in the paper (Fig. 2) but the scaling of the speedup remains qualitatively similar (α ~ 0.5).

a


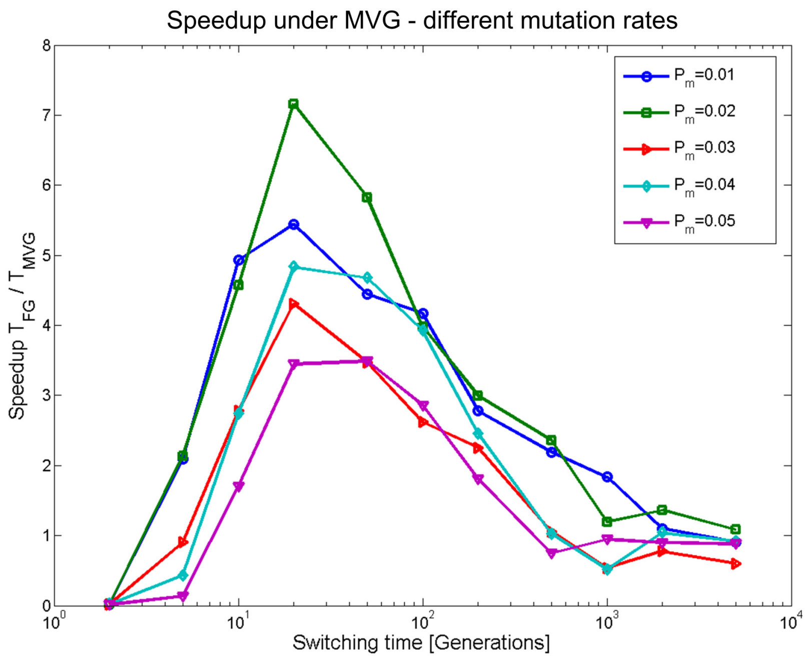
b


Fig. 20. Effect of evolutionary parameters on speedup. (*a*) Speedup (S = T_{FG}/T_{MVG}) under different frequencies of goal switches and with various population size (N_{pop}). Selection was pure fitness based. Mutation rate was $P_m = 0.7$ per genome per generation. (*b*) Different mutation rates (P_m here is per bit per generation). $N_{pop} = 100$. Results are shown for goal G1 = (*x* XOR *y*) OR (*w* XOR *z*) using model 2. For MVG

the varying goal switched between G1 and G2 = (x XOR y) AND (w XOR z) every E generations.

7. Speedup under MVG using a hill climbing algorithm

We find that a speedup occurs under MVG also using different algorithms (other than genetic algorithms). Here we present results of a simple probabilistic hill-climbing algorithm (6) described below. We find that the speedup scales with problem difficulty with an exponent $\alpha \sim 0.7$ in various scenarios of modularly varying goals (SI Fig. 21).

Hill climbing algorithm description. Population size equals 1 (a single circuit). At each generation, the fitness of all possible B mutations (neighbors) is calculated (denoted F_i). Then one mutation is randomly chosen according to the following distribution (as in Metropolis Monte Carlo algorithms (8)): $P = exp(F_i/t) / Z$, where here Z is a normalizing factor (so that the sum of all P equals 1). The parameter t, is referred to as the temperature. For very large t, this becomes a random walk (as all mutations have equal probability). For very small t, it becomes deterministic hill-climbing as the mutation with the highest fitness is always chosen.

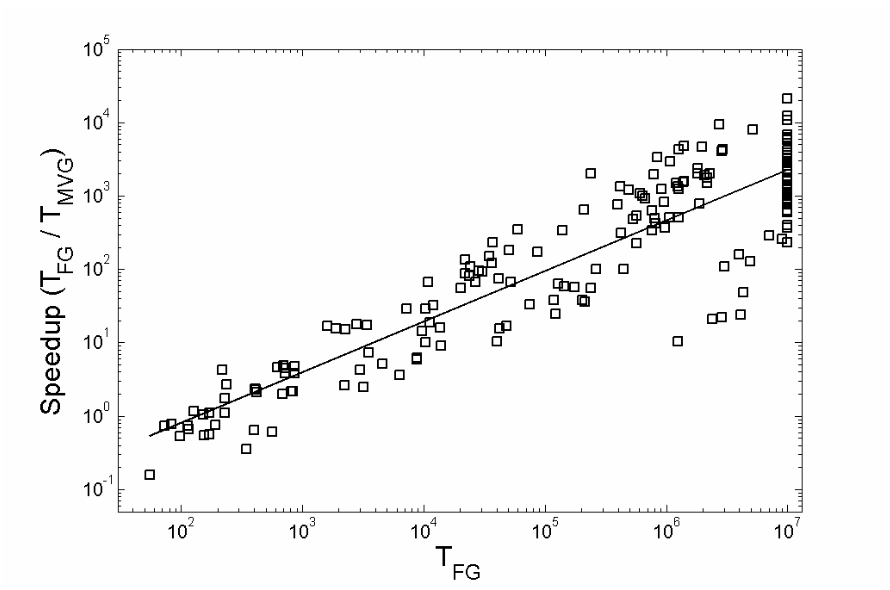


Fig. 21. Speedup under MVG using a hill-climbing algorithm. Each point represents a speedup of a different goal. Goals were various 4-input 1-output Boolean functions (model 2, small version). We chose goals that could be decomposed into two 2-input 1-output Boolean functions (denoted by F1, F2). For MVG the goal was switched every E = 20 generations between G1 = F1 AND F2 and G2 = F1 OR F2. Speedup scales as a power law with exponent $\alpha = 0.7$. Results shown for $T = 0.03$.

1. Vasconcelos JA, Ramirez JA, Takahashi RHC, Saldanha RR (2001) *Proc IEEE Trans Magnet*, Vol 37:3414-3418.
2. Kashtan N, Alon U (2005) *Proc Natl Acad Sci USA* 102:13773-8.
3. Alon U (2006) *An Introduction to Systems Biology: Design Principles of Biological Circuits* (Chapman & Hall/CRC, Boca Raton, Florida).
4. Hofacker IV, Fontana W, Stadler PF, Bonhoeffer LS, Tacker M, Schuster P (1994) *Monatsh Chem* 125:167-188.
5. Goldberg D (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley Publishing Company, Reading, MA).
6. Johnson AW, Jacobson SH (2002) *Discrete Applied Mathematics* 119:37-57.
7. Bray D (1995) *Nature* 376**,** 307-12.
8. Newman MEJ, Barkema GT (1999) *Monte Carlo Methods in Statistical Physics* (Oxford University Press).
9. Itzkovitz S, Tlusty T, Alon U (2006) *BMC Genomics.* 7 (1):239.
10. Mitchell, M (1996) *An Introduction to Genetic Algorithms* (MIT Press, Cambridge MA).
11. Kim SH , Suddath FL, Quigley GJ, McPherson A, Sussman JL, Wang AHJ, Seeman NC and Rich A (1974), *Science* 185:435-440.